ECE 403
Senior Design II
FPGASolve
Options Considered Document
9/21/09

Swati Gupta
Richard Schultz
Matthew Nitschke

Introduction:

The FPGASolve project is a hardware accelerator for use with the Newton-Raphson Method of power flow analysis. Ideally it will dramatically decrease the time that software takes to solve the system of equations used in the Newton-Raphson method. This project was proposed by Professor Ababei and Professor Kavasseri for the purpose of a more efficient optimal network reconfiguration.

Previous Work:

Parallel Solutions of Newton's Power Flow Equations on Configurable Chips is a similar project that was carried out at Drexel University by Xiaofang Wang, Sotirious Ziavras, Chika Nwankpa, Jeremy Johnson, and Prawat Nagvarjara. They were successful at using FPGAs to speed up their computations by an order of magnitude compared to the traditional software method. We plan on using their research as a basis for the FPGASolve project.

Design Options and Selected Approach:

Software Coding Options: C / MATLAB

C is an almost universal coding software that is widely used. It is also user friendly and relatively easy to edit. MATLAB is a very powerful mathematical problem solving tool that has many mathematical function built into the program. It also has the benefit that most of the FPGASolve design team has experience using MATLAB.

Decision: We have decided to use MATLAB based on the fact that we have the most experience with this programming platform and feel that it will be easier to implement the complex mathematical functions necessary for this project than it would be using C/C++.

FPGA Options: Xilinx / Altera

There are many FPGA options available but we have limited our choice to Xilinx and Altera due to the fact that they are the most readily available to use and the most popular in the industry.

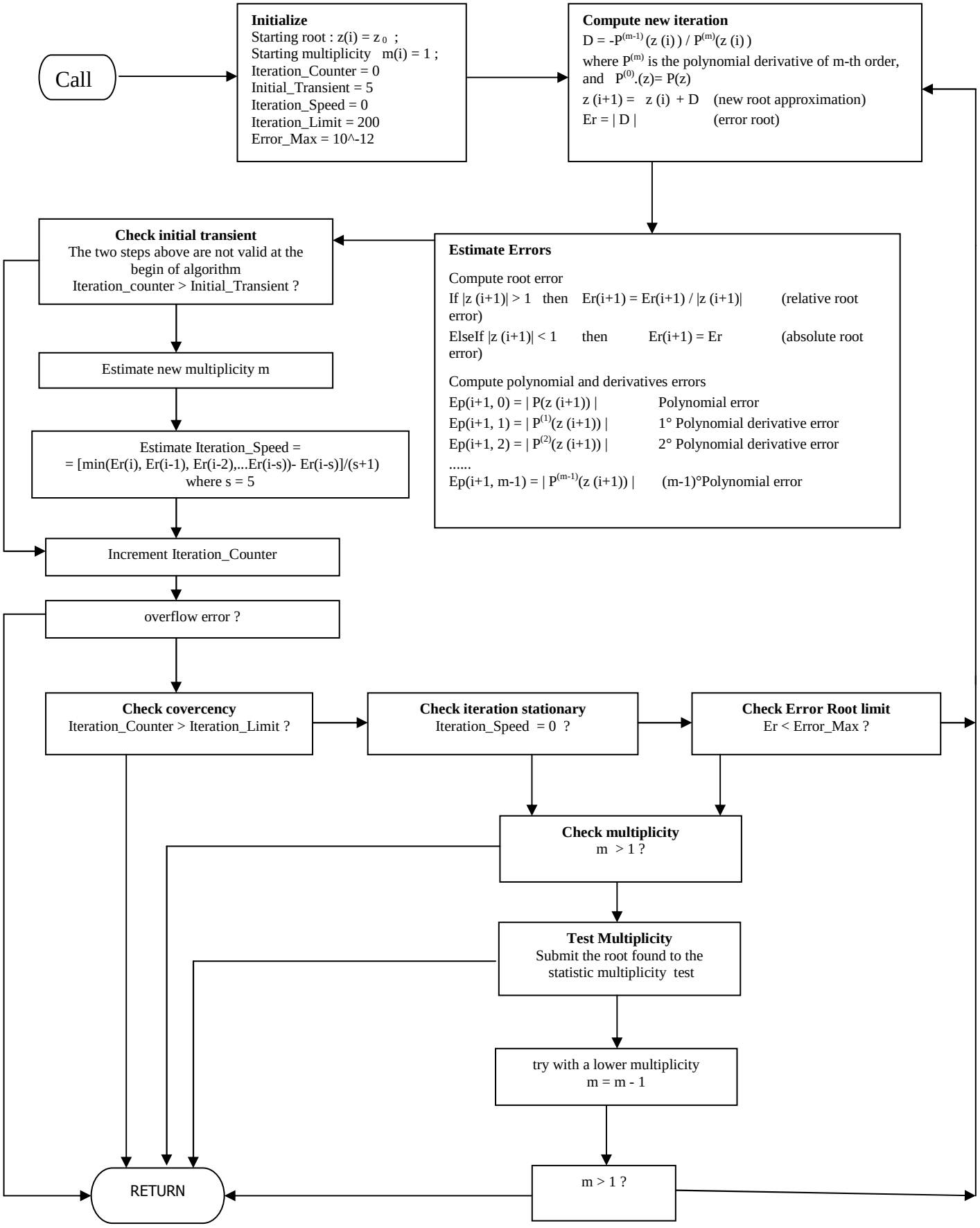
Decision: We chose Altera at the suggestion of our advisor who has used both types FPGAs and found that the Altera board is the easier to use.

Hardware Coding: C / VHDL

There are many options that we have to choose from for programming our FPGA. Since we have chosen the Altera FPGA, we could use Altera's C2H compiler that takes C and compiles it into VHDL that the FPGA needs to operate. The advantage of this is that our design team has had some background with C and no experience with VHDL. The disadvantage of this is that the Altera compiler is not cheap and would make our budget very high. Our other option is to learn VHDL and implement the code in VHDL. VHDL is the hardware description language that will allow us to code at the base level.

Decision: At the request of our advisor we have chosen to code using VHDL. By choosing this option our design team will obtain a very in depth knowledge of how to code using VHDL which is the basis for coding hardware.

Flowchart of Newton Raphson's Method :


Call
Initialize
Starting root : z(i) = z 0 ;
Starting multiplicity m(i) = 1 ;
Iteration_Counter = 0
Initial_Transient = 5
Iteration_Speed = 0
Iteration_Limit = 200
Error_Max = 10^-12
Compute new iteration
D = -P(m-1) (z (i)) / P(m)(z (i))
where P(m) is the polynomial derivative of m-th order, and P(0).(z)= P(z)
z (i+1) = z (i) + D (new root approximation)
Er = | D | (error root)
Estimate Errors
Compute root error
If |z (i+1)| > 1 then Er(i+1) = Er(i+1) / |z (i+1)| (relative root error)
ElseIf |z (i+1)| < 1 then Er(i+1) = Er (absolute root error)
Compute polynomial and derivatives errors
Ep(i+1, 0) = | P(z (i+1)) | Polynomial error
Ep(i+1, 1) = | P(1)(z (i+1)) | 1° Polynomial derivative error
Ep(i+1, 2) = | P(2)(z (i+1)) | 2° Polynomial derivative error
.....
Ep(i+1, m-1) = | P(m-1)(z (i+1)) | (m-1)°Polynomial error
Check initial transient
The two steps above are not valid at the begin of algorithm
Iteration_counter > Initial_Transient ?
Estimate new multiplicity m
Estimate Iteration_Speed =
= [min(Er(i), Er(i-1), Er(i-2),...Er(i-s))- Er(i-s)]/(s+1)
where s = 5
Increment Iteration_Counter
overflow error ?
Check covercency
Iteration_Counter > Iteration_Limit ?
Check iteration stationary
Iteration_Speed = 0 ?
Check Error Root limit
Er < Error_Max ?
Check multiplicity
m > 1 ?
Test Multiplicity
Submit the root found to the statistic multiplicity test
try with a lower multiplicity
m = m - 1
m > 1 ?
RETURN

Budget:

Table 1: Budget

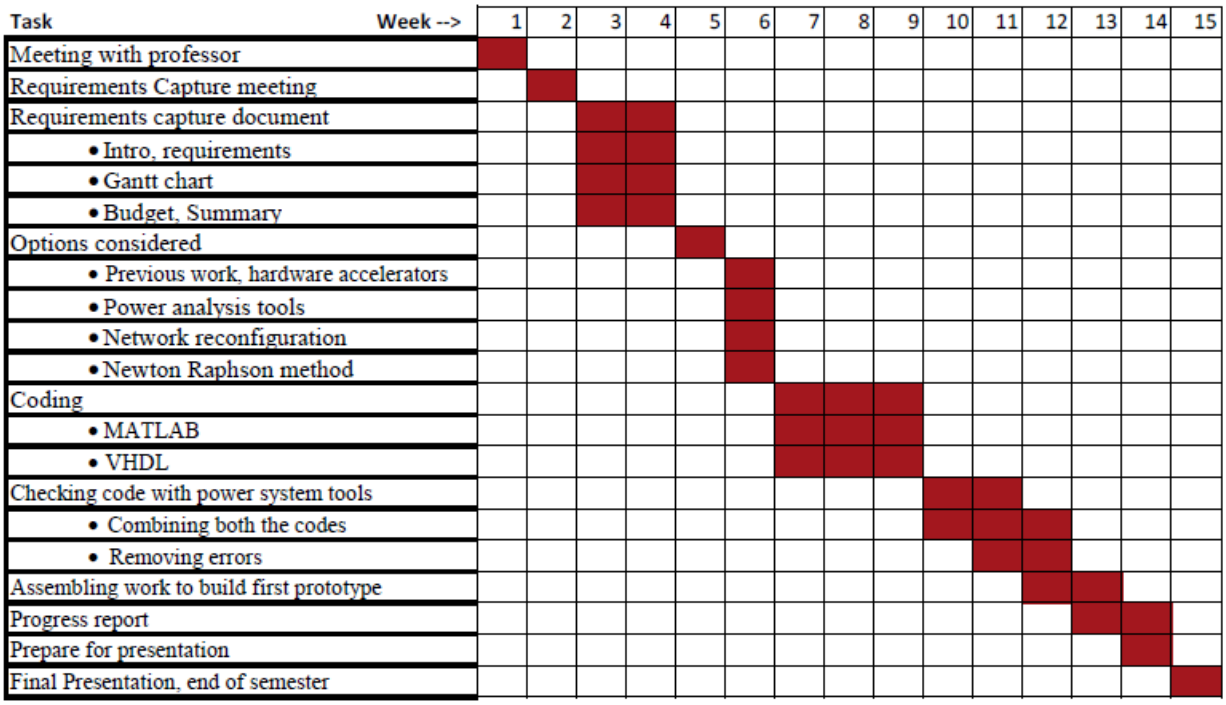
Part	Quantity	Retail Cost	Expected Cost	Total Cost	Notes
Altera FPGA Development Board	2	$150	$0	$0	1 Board provided by Advisor
Hadi Sadat Power System Analysis book	1	$160	$160	$160	
MATLAB License	2	$100	$100	$200	
Misc.	1		$72	$72	20% of total costs
			Total	$432	

Timeline :

Name	**Weeks**	**Who**
Meeting with professor	1	All
Requirements Capture meeting	2	All
Requirements capture document	3,4	All
• Intro, requirements	3,4	Matt
• Gantt chart	3,4	Swati
• Budget, Summary	3,4	Richard
Options considered	5	All
• Previous work, hardware accelerators	6	All
• Power analysis tools	6	Matt
• Network reconfiguration	6	Richard
• Newton Raphson method	6	Swati
Coding	7,8,9	All
• MATLAB	7,8,9	Richard, Matt
• VHDL	7,8,9	Swati
Checking code with power system tools	10,11	All
• Combining both the codes	10,11,12	All
• Removing errors	11,12	All

Assembling work to build first prototype	12,13	All
Progress report	13,14	All
Prepare for presentation	14	All
Final Presentation, end of semester	15	All

Gantt Chart :



Summary: We have considered all of the requirements and options carefully while putting this document together, and we feel that we have made the most reasonable and efficient choices for our project. The goal of our project is to create a hardware accelerator using FPGA hardware that will greatly increase the efficiency of computing power flow solutions with the Newton-Raphson method. In our budget, we have outlined the materials and costs necessary for our project to succeed. As indicated in our timeline, we plan to have a working prototype of the top-level MATLAB software design and the VHDL coded FPGA hardware accelerator completed by the end of this semester. We will then spend next semester refining and optimizing our design to make the system as efficient as possible. The FPGA hardware accelerator that we plan to create will make analyzing power systems much more efficient and will make optimal network reconfiguration a faster and simpler task.